// Copyright 2022 The Sigstore Authors.

//

// Licensed under the Apache License, Version 2.0 (the "License");

// you may not use this file except in compliance with the License.

// You may obtain a copy of the License at

//

// http://www.apache.org/licenses/LICENSE-2.0

//

// Unless required by applicable law or agreed to in writing, software

// distributed under the License is distributed on an "AS IS" BASIS,

// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

// See the License for the specific language governing permissions and

// limitations under the License.

syntax = "proto3";

package dev.sigstore.bundle.v1;

import "google/api/field\_behavior.proto";

// https://raw.githubusercontent.com/secure-systems-lab/dsse/9c813476bd36de70a5738c72e784f123ecea16af/envelope.proto

import "envelope.proto";

import "sigstore\_common.proto";

import "sigstore\_rekor.proto";

option go\_package = "github.com/sigstore/protobuf-specs/gen/pb-go/bundle/v1";

option java\_package = "dev.sigstore.proto.bundle.v1";

option java\_multiple\_files = true;

option java\_outer\_classname = "BundleProto";

option ruby\_package = "Sigstore::Bundle::V1";

// Notes on versioning.

// The primary message ('Bundle') MUST be versioned, by populating the

// 'media\_type' field. Semver-ish (only major/minor versions) scheme MUST

// be used. The current version as specified by this file is:

// application/vnd.dev.sigstore.bundle.v0.3+json

// The semantic version is thus '0.3'.

// Various timestamped counter signatures over the artifacts signature.

// Currently only RFC3161 signatures are provided. More formats may be added

// in the future.

message TimestampVerificationData {

// A list of RFC3161 signed timestamps provided by the user.

// This can be used when the entry has not been stored on a

// transparency log, or in conjunction for a stronger trust model.

// Clients MUST verify the hashed message in the message imprint

// against the signature in the bundle.

repeated dev.sigstore.common.v1.RFC3161SignedTimestamp rfc3161\_timestamps = 1;

}

// VerificationMaterial captures details on the materials used to verify

// signatures. This message may be embedded in a DSSE envelope as a signature

// extension. Specifically, the `ext` field of the extension will expect this

// message when the signature extension is for Sigstore. This is identified by

// the `kind` field in the extension, which must be set to

// application/vnd.dev.sigstore.verificationmaterial;version=0.1 for Sigstore.

// When used as a DSSE extension, if the `public\_key` field is used to indicate

// the key identifier, it MUST match the `keyid` field of the signature the

// extension is attached to.

message VerificationMaterial {

// The key material for verification purposes.

//

// This allows key material to be conveyed in one of three forms:

//

// 1. An unspecified public key identifier, for retrieving a key

// from an out-of-band mechanism (such as a keyring);

//

// 2. A sequence of one or more X.509 certificates, of which the first member

// MUST be a leaf certificate conveying the signing key. Subsequent members

// SHOULD be in issuing order, meaning that `n + 1` should be an issuer for `n`.

//

// Signers MUST NOT include root CA certificates in bundles, and SHOULD NOT

// include intermediate CA certificates that appear in an independent root of trust

// (such as the Public Good Instance's trusted root).

//

// Verifiers MUST validate the chain carefully to ensure that it chains up

// to a CA certificate that they independently trust. Verifiers SHOULD

// handle old or non-complying bundles that have superfluous intermediate and/or

// root CA certificates by either ignoring them or explicitly considering them

// untrusted for the purposes of chain building.

//

// 3. A single X.509 certificate, which MUST be a leaf certificate conveying

// the signing key.

//

// When used with the Public Good Instance (PGI) of Sigstore for "keyless" signing

// via Fulcio, form (1) MUST NOT be used, regardless of bundle version. Form (1)

// MAY be used with the PGI for self-managed keys.

//

// When used in a `0.1` or `0.2` bundle with the PGI and "keyless" signing,

// form (2) MUST be used.

//

// When used in a `0.3` bundle with the PGI and "keyless" signing,

// form (3) MUST be used.

oneof content {

dev.sigstore.common.v1.PublicKeyIdentifier public\_key = 1 [(google.api.field\_behavior) = REQUIRED];

dev.sigstore.common.v1.X509CertificateChain x509\_certificate\_chain = 2 [(google.api.field\_behavior) = REQUIRED];

dev.sigstore.common.v1.X509Certificate certificate = 5 [(google.api.field\_behavior) = REQUIRED];

}

// An inclusion proof and an optional signed timestamp from the log.

// Client verification libraries MAY provide an option to support v0.1

// bundles for backwards compatibility, which may contain an inclusion

// promise and not an inclusion proof. In this case, the client MUST

// validate the promise.

// Verifiers SHOULD NOT allow v0.1 bundles if they're used in an

// ecosystem which never produced them.

repeated dev.sigstore.rekor.v1.TransparencyLogEntry tlog\_entries = 3;

// Timestamp may also come from

// tlog\_entries.inclusion\_promise.signed\_entry\_timestamp.

TimestampVerificationData timestamp\_verification\_data = 4;

}

message Bundle {

// MUST be application/vnd.dev.sigstore.bundle.v0.3+json when

// when encoded as JSON.

// Clients must to be able to accept media type using the previously

// defined formats:

// \* application/vnd.dev.sigstore.bundle+json;version=0.1

// \* application/vnd.dev.sigstore.bundle+json;version=0.2

// \* application/vnd.dev.sigstore.bundle+json;version=0.3

string media\_type = 1;

// When a signer is identified by a X.509 certificate, a verifier MUST

// verify that the signature was computed at the time the certificate

// was valid as described in the Sigstore client spec: "Verification

// using a Bundle".

// <https://docs.google.com/document/d/1kbhK2qyPPk8SLavHzYSDM8-Ueul9\_oxIMVFuWMWKz0E/edit#heading=h.x8bduppe89ln>

// If the verification material contains a public key identifier

// (key hint) and the `content` is a DSSE envelope, the key hints

// MUST be exactly the same in the verification material and in the

// DSSE envelope.

VerificationMaterial verification\_material = 2 [(google.api.field\_behavior) = REQUIRED];

oneof content {

dev.sigstore.common.v1.MessageSignature message\_signature = 3 [(google.api.field\_behavior) = REQUIRED];

// A DSSE envelope can contain arbitrary payloads.

// Verifiers must verify that the payload type is a

// supported and expected type. This is part of the DSSE

// protocol which is defined here:

// <https://github.com/secure-systems-lab/dsse/blob/master/protocol.md>

// DSSE envelopes in a bundle MUST have exactly one signture.

// This is a limitation from the DSSE spec, as it can contain

// multiple signatures. There are two primary reasons:

// 1. It simplfies the verification logic and policy

// 2. The bundle (currently) can only contain a single

// instance of the required verification materials

// During verification a client MUST reject an envelope if

// the number of signatures is not equal to one.

io.intoto.Envelope dsse\_envelope = 4 [(google.api.field\_behavior) = REQUIRED];

}

// Reserved for future additions of artifact types.

reserved 5 to 50;

}